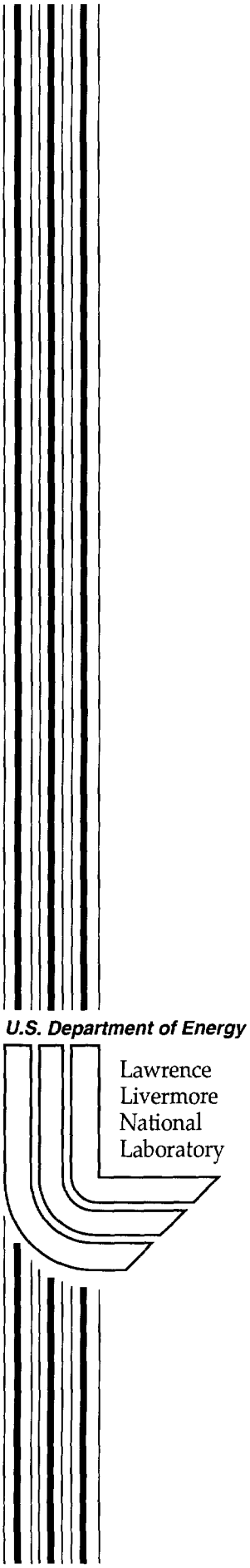
Preprint
UCRL-JC-147099

Structured Surface Grid Generation on Boundary Represented Geometry

J. J. Chou

This article was submitted to
8th International Conference on Numerical Grid Generation in Computational Field Simulations, Honolulu, HI., June 2-6, 2002

January 18, 2002



This work was performed under the auspices of the United States Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Structured Surface Grid Generation On Boundary Represented Geometry

Jin J. Chou

Lawrence Livermore National Laboratory
University of California
Livermore, CA 94550
chou6@llnl.gov

Abstract

Generation of surface meshes is the first step in many grid generation processes. For the generation of block-structured meshes, structured surface meshes have to be generated first. This paper investigates the problem of generating a structured surface mesh across multiple surface patches on an object with the boundary representation and relates the problem to other commonly encountered issues in CAD/CAM. It describes a method for solving the problem. This method is based on initial surface construction, point projection and a mixed model-space and parameter-space based elliptic smoothing.

Introduction

Despite the improvements in unstructured and semi-structured (e.g., prism meshes) grid generation methods and advancements in solvers technology, many applications still require body-conforming structured meshes with finely controlled point distribution in order to obtain accurate solutions [1]. A structured mesh requires a topologically rectangular array of grid points. Self-folding meshes or unsmooth meshes are frequently the root causes of solver divergence or inaccurate solutions. The required level of smoothness to obtain accurate solutions is highly dependent on the problem and solver at hand; however, self-folding meshes or meshes with negative Jacobian are definitely undesirable.

Traditionally, structured surface mesh generation is restricted to the requirement of a surface mesh completely lying on a single CAD surface [2][3]. However, due to the popularity of CAD design systems operated on the basis of boundary represented solids and Boolean operations, design models produced by CAD systems tend to bear the history of the sequence of CAD operations producing the design, with surfaces broken into pieces resulting from the Boolean operations [4]. A logical structured surface mesh for an analysis is frequently required to straddle over multiple CAD faces, each being a trimmed portion of a CAD surface. These CAD faces may or may not have the same underlying (untrimmed) surface representations.

Even if some of the faces have the same surface representations in model space, they may have different parametrizations as presented to the grid generators.

Restricting a surface mesh to completely lie on a single CAD surface at the least increases the number of mesh blocks required to complete the blocking of a configuration for an analysis; at worst, it makes the blocking impossible. It also impacts the quality of the meshes since many CAD models contain faces of irregular shapes and various sizes. Frequently faces may have sizes that are smaller than the desired grid sizes but are too big to be ignored. Figure 1 shows the bottom surface of a CAD model of the NASA X38 Crew Return Vehicle (CRV). The highlighted lines are edges between faces. We can see that even though the bottom portion of the CRV is flat and has a regular shape as a whole, it contains many faces of various sizes, and shapes, and some of them contain degeneracies. It is highly desirable to be able to handle the whole bottom portion in one mesh (as shown in Figure 2) for analysis.

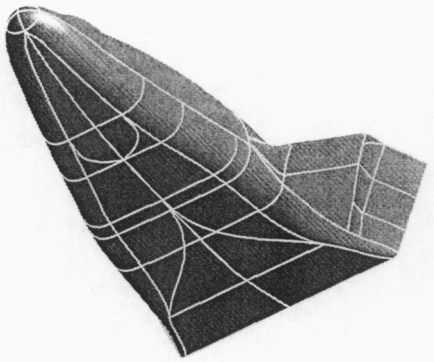


Figure 1 The Crew Return Vehicle.

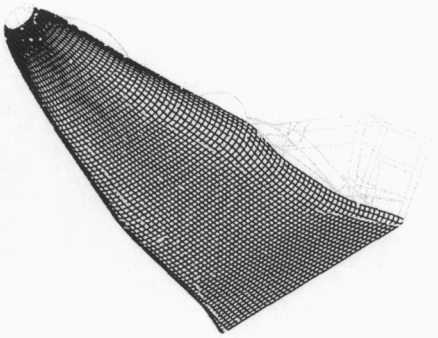


Figure 2 A Mesh for the Bottom of the Crew Return Vehicle.

To solve the problem of spanning a smooth mesh over multiple faces, each with their own parametrization, we can begin by searching for ways to construct a single smooth surface with a single uniform parametrization laid over the set of faces. One method commonly used in CAD is sample-and-fit. That is, a two dimensional array of points is sampled from the faces and then a tensor product spline surface is fit to these points to approximate the faces [5]. However, to find a good set of fitting points to produce a smooth surface with uniform parametrization is difficult. To a certain extent, a good set of fitting points may even be used to substitute for the structured mesh that we are seeking. Conversely, if a uniformly distributed structured surface mesh is given, most trivial fitting algorithms can produce a good spline surface to approximate the underlying faces. This leads the author to conclude that the problem of finding a single uniformly parametrized surface fitting over a set

of faces is equivalent, in difficulty, to the construction of a smooth structured surface mesh. Just as a smooth structured surface mesh is important to certain computational physics, the generation of a good spline surface fitting to a set of faces has many other applications, which will be explored more at the end of this paper.

In this paper, a method utilizing a combination of CAD and grid generation techniques is presented to solve the problem of generating a smooth structured mesh over a set of faces connected with topology. To be precise, the faces are part of a single boundary representation shell, either open or closed [4]. If the shell is closed, it may be a part of a solid model.

Outline of the Method

Given a set of faces and four lists of edges prescribing the four boundaries of the mesh, we generate a surface mesh through the following steps.

1. Create a single surface that approximates the faces and conforms to the prescribed mesh boundary. This surface produces a very crude approximation and may even be self-folding.
2. An initial mesh is generated from this surface by any traditional surface mesh generation method. This initial mesh, most likely, is not smooth since the underlying surface is not smooth.
3. The initial mesh is projected onto the set of given faces. This projected mesh, again, is not likely to be smooth.
4. An elliptic smoothing step is applied to the projected initial mesh on the parameter spaces of the given faces.

The final mesh is a smooth mesh on the given faces conforming to the given faces and boundary edges. If mesh clustering controls are desired, they are applied in both the initial mesh generation and the final elliptic smoothing. Details of the method are described in the following sections.

Approximation Surface Generation And Initial Mesh Generation

As mentioned above, traditional structured mesh generators require a single parametrization; however, generating a *smoothly* parametrized surface from a set of faces is as difficult as generating a smooth mesh. In this step, we relax the smoothness requirement of the surface. Instead, we construct *one* tensor product spline surface. We require this surface to reproduce the boundaries of the mesh, given as lists of edges. First we construct the four boundaries curves from the edges. A boundary curve is constructed by concatenating together the curves from the edges in a boundary. The parametrization of the boundary curves affects the created spline

surface greatly. If there are points on the opposite boundaries that the user has identified as corresponding to each other, we reparametrize the boundary curves to have identical parameter at those points. The method in [6] is used to construct the approximating surface from the four boundary curves. Note that since the constructed surface is a kind of Coon's patch, the interior of this surface is, in general, not conforming to the interior region of the given faces and the surface may "overspill".

The initial mesh is generated on this surface with algebraic methods. Any of a number of the existing methods can be used. We use a method similar to that in [7]. First, grid points are distributed on the boundaries of the surface with user specified grid clustering control requirements. These grid points on the boundaries are in their final positions in the mesh, since the surface boundary reproduces the mesh boundary exactly. When we perform this grid distribution, we also obtain the face parameter u and v values at these boundary points. The interior grid points are obtained as done in [7]. The sole purpose of the Coon's patch is to produce this initial mesh. It is not used after this stage.

We then project the interior initial grid points onto the given faces. This is done by marching along each of the grid lines from boundaries inwards and projecting each point along the way in sequence. Two types of projection can be employed. One is to find the closest point. The other is to find the intersection between the faces and a line passing through the given grid point being projected. In the latter method, a possible line of intersection is a line along the average mesh normal on the initial mesh at the grid point. However, since the initial mesh may contain folds and overspills, the mesh normal is not reliable. So we choose the closest point approach.

Either local search or global search can be used to find the closest point. In local search, an initial guess point is given, and iterative methods (e.g., Newton method) are used to close in the exact solution. However, iterative methods may not converge. Even they do, they may converge to a local minimum. Global search will hunt for a set of possible candidate solutions by looking through the whole surface domain. From this set of possible candidate solutions, local searches are carried out to find the closest point. Obviously local search is much cheaper computationally. When we project the interior initial grid points, we move from the boundary points to the mesh center along each of the grid lines. For each point, a local search is first used with initial guess point from the neighboring (or boundary points for the first inner layer of grids away from the boundary) projected points. If this search fails to converge or the distance to the closest point found is too far from those distances found for the projected neighboring points, we perform a global search.

Since we project onto faces, which are trimmed portions of surfaces, on every Newton iteration, we need to detect the potential crossing from one face to another. We do this by computing the intersection of the face boundary with the line segment

formed between the point obtained in the current iteration and the point from the previous iteration. This computation is performed in the parameter space of the surface using the parameter space trimming loops of the current face.

If the crossing happens, two possible situations arise. If we cross into a face that is not one of the given faces, we stop at the boundary. This may happen when the approximation surface overspills outside the mesh boundary. A global search may be prescribed to ensure that we are not wandering outside of the mesh boundary due to convergence to a wrong local minimum.

If we cross into a face in the given face set, we need to determine the parameter value at the crossing point on the new face. To help with this, we require the two parameter space trimming curves of an edge on the two faces to have rough parametric correspondence. That is, given two faces F_1 and F_2, with surfaces S_1 and S_2, respectively, sharing an edge E with parameter space curves P_1 and P_2, on S_1 and S_2, respectively, we require that

1. both P_1 and P_2 have the same parameter range, say, [0,1] and
2. $S_1(P_1(t^*))$ is very close to $S_2(P_2(1-t^*))$, for any t^* in [0,1].

Note that the two parameter space curves go in the opposite directions. In the face crossing detection, we intersect the parameter space trimming loops of the current face with the line segment connecting points between the previous and the next iteration. In the intersecting process, we identify the edge, the parameter space curve, and the curve's parameter value at the intersection point. From this information and the parameter correspondence condition listed above, we can determine into which face the new iteration is crossing and the parameter value of the face at the crossed point. The iterative local search for the minimum distance point will continue from the new face and the new initial guess point. From the projection, for each of the grid points we obtain the face that the point is on and both the model space and parameter space locations of the projected point.

Elliptic Smoothing

The heart of the method is an elliptic solver which untangles the initially projected mesh and imparts the required mesh clustering. Our elliptic grid smoother is very similar to that in [3], which re-iterates the equations in Chapter VI of [8]. Here, we do not go into the derivations of the equations in detail. Instead, we simply list the equations that we used. Readers who need more background on elliptic grid generation should consult those two sources.

Two sets of solvers are used in our smoother. One solves the set of elliptic equations in model space. The other solves the same problem with a different set of equations in the parameter space.

The 3D model space equation for elliptic surface grid generation is [8][9]

$$\alpha(r_{\xi\xi}+\Phi r_{\xi}) - 2\beta r_{\xi\eta} + \gamma(r_{\eta\eta}+\Psi r_{\eta}) = [(\alpha r_{\xi\xi} - 2\beta r_{\xi\eta} + \gamma r_{\eta\eta}) \cdot n]n, \quad (1)$$

where r = (x,y,z) is the physical space point; (ξ,η) are the computational coordinates for the surface; $\alpha = r_{\xi} \cdot r_{\xi}$, $\beta = r_{\xi} \cdot r_{\eta}$, $\gamma = r_{\eta} \cdot r_{\eta}$; n is the unit normal of the surface at the grid point; Φ and Ψ are the control functions for grid spacing control.

For solving equation (1) in parameter space, we introduce the surface parameter point w = (u,v) = $(u(\xi,\eta),v(\xi,\eta))$ and rewrite the surface equation as r = (x(u,v), y(u,v), z(u,v)). The model space equation (1) can be transformed to

$$\alpha(w_{\xi\xi}+\Phi w_{\xi}) - 2\beta w_{\xi\eta} + \gamma(w_{\eta\eta}+\Psi w_{\eta}) = J(\alpha w_{\xi}/J - \beta w_{\eta}/J)_{\xi} + J(-\beta w_{\xi}/J + \gamma w_{\eta}/J)_{\eta}, \quad (2)$$

where $J = |r_{\xi} \times r_{\eta}|$ is the Jacobian. Readers who are interested in the derivation of the above equation should consult [8].

Equations (1) and (2) are solved numerically with iterative methods by writing the equations into a set of difference equations. Most commonly used methods are line-sweep and point-wise iterations [10].

The disadvantage of solving the model space equation (1) is that the resulting points may not lie on the face. This problem can be handled by projecting the solution obtained from the difference equations back onto the faces. This projection is done after every iteration during the difference equations solving process. This projection is similar to the projection of initial mesh points as described in the previous section. For the same reason as stated in that section, we choose closest point finding as the projection method. Note that this projection may also move a point across face boundary into a different face. So the line segment joining the two points before and after each iteration in the projection is intersected with the trimming loops of the faces to detect and determine face crossing. Since we know the face that each point is on and the face parameter values at such point, we use local search for this closest point finding. In case it fails, then a global search is performed.

The advantage of the parametric space equation is that the equation solves for the u and v values of the grid points. The grid points thus obtained are guaranteed to be on the surfaces. But there are two restrictions on the usage of the equation. First, in order for the equation to apply, the neighboring points used to compute the differences in obtaining the numerical solutions must be on the same face. We use central differencing for interior grid points and one-sided differencing for boundary grid points. All those points involved in the differencing should be on the same face. Second, in order for the numerical solution to converge, the surfaces underlying the faces must not contain discontinuities. This is a potential problem for spline

surfaces, which may not have second order derivative continuity. To cope with this problem, for each grid point, we also keep a record of which subpatch of the surface the point is on. A subpatch is defined as the derivative continuous subdomain of the spline surface.

We use both equations (1) and (2) in our mesh smoothing. Point-wise iteration is used with alternate sweeping in both i and j directions. For each grid point, we look at whether all the relevant neighboring points in the difference equations are on the same face and on the same subpatch. If they do, the restrictions on solving the parameter space equation are satisfied; we solve the parameter space equation for this point. Otherwise, we solve the model space equation, which is then followed by a projection.

Note that although the parametric space equation guarantees the resulting grid points remain on the surfaces underlying the faces, a point may move outside the trimming loops of the face. That would result in the grid point lying outside the face. We handle this problem by detecting the migration of the point across face boundary. Detecting the migration is very similar to the detection of the crossing of faces during point projection in the previous section. On every point iteration of the difference equation solver, we intersect the line segment connecting the point's locations before and after the iteration with the face's loops. (This intersection is done in the parameter space of the surface.) If an intersection is detected, the point is migrating across to a different face. When the crossing happens, we have to update the point's face, its u, v values and its subpatch. The likelihood for such migration to happen is low if the trimming loops are smooth. But the computational expense is high for detecting such migration since it has to be done for every point on every iteration. We save computation time by turning on point migration detection only when we have convergence problems in our solver.

Sample Results

In this section, we show three examples of the application of the above method on mesh generation. Figure 3 shows two arms of a portion of a solid part. We want to generate structured meshes on its outside surface. Here we show the process of generating one of the surface meshes. Figure 4 shows the generated final mesh overlaid on the solid part. This mesh covers a domain of two faces: the top outside face of the vertical arm and the top fillet face where this arm is connected to its base block.

To generate the mesh, first an approximating surface is created from the four boundaries (Figure 5). It is apparent that this surface is a poor approximation to the desired faces. An initial mesh is generated on this surface and projected onto the

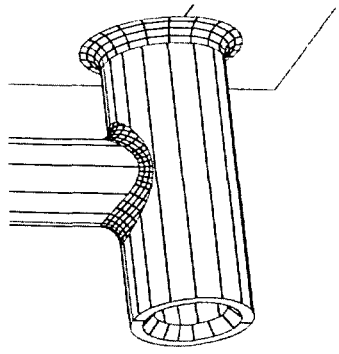


Figure 3 Two Arms of a Solid Part.

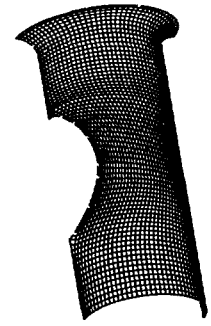


Figure 4 Final Smoothed Mesh.

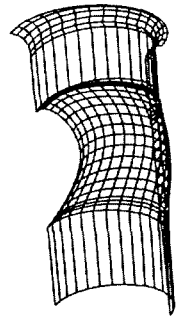


Figure 5 The Spline Surface Approximating the Faces.

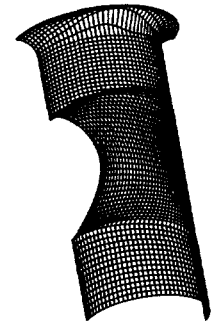


Figure 6 Initial Mesh Projected on the Faces.

two underlying faces. The projected mesh is shown in Figure 6. This projected mesh is not a smooth mesh. It contains folds in many locations. Its unsmoothness and folds come from the fact that the approximation surface is skewed and is lifted away from the faces due to the method that we used to obtain this approximation surface. The elliptic smoother described above was applied to this projected mesh on the faces. After 20 iterations, the final mesh comes to the desired smoothness. All the grid points on the mesh lie exactly on the faces (Figure 4).

The second example is the CRV shown in the Introduction. This example demonstrates the idea of covering a large set of CAD faces with one surface mesh. We used our method to generate a mesh for the 33 faces at the bottom portion of the CRV. Twenty iterations were performed by the smoother to achieve the shown

smoothness (Figure 2). It took 4 seconds on a PC with 950 MHz Pentium CPU to generate the smoothed grid of 3200 (80X40) points, including approximation surface creation, initial grid generation, grid projection and elliptic smoothing.

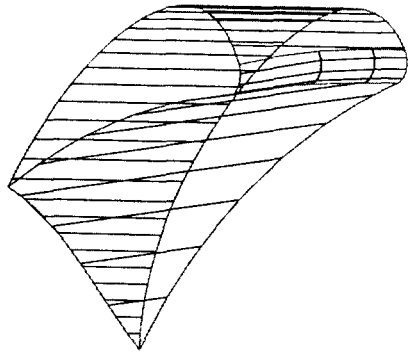


Figure 7 An Airfoil Shape.

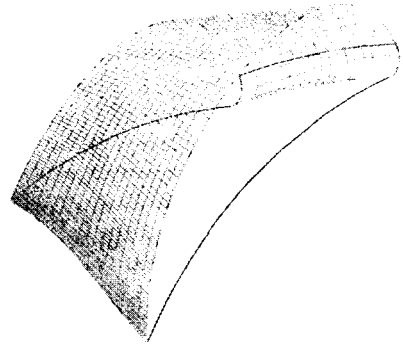


Figure 8 Smoothed Mesh.

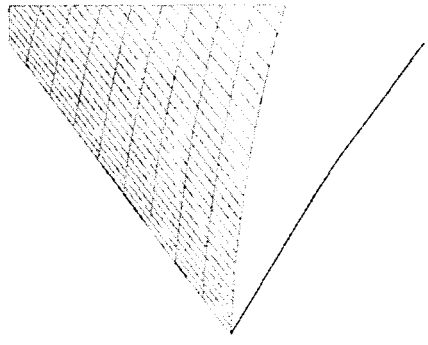


Figure 9 Mesh Near Trailing Edge.

The last example is an airfoil shape (Figure 7). We generate a surface mesh covering the top face and the face in the leading edge. Figure 8 shows the final smoothed mesh. In the figure, we can see a horizontal grid line goes across the edge joining these two faces, and the mesh is smooth around the crossing. This mesh is generated with extreme grid clustering near the trailing edge of the airfoil. Figure 9 is a zoom-in view of the mesh around the trailing edge. The mesh maintains very good quality even with the extreme clustering.

Conclusion

Generation of a structured surface mesh across multiple faces is a difficult task but with high potential payback. We have demonstrated a method that combines surface reconstruction, projection and elliptic smoothing for this purpose. Also as it is pointed out earlier in the Introduction, the problem of generating a smooth structured surface mesh over a set of faces is related to refitting a set of faces with a tensor product surface, which in turn has many other applications, for example, obtaining a unified (u,v) parametrization for points on a set of faces, and generation of numerical

controlled (NC) tool paths to cover a given set of faces. For obtaining a unified parametrization, a reasonably dense and smooth mesh can first be generated on the faces by our method. Any point on the faces can be assigned a (u,v) parametrization by either referring to the (u,v) of the closest point on the refitted tensor product surface or by simply interpolating the (u,v) from the corners of the neighboring mesh points.

Various mesh clustering strategies can be applied to control the spacing of the mesh points. In NC machine tool path generation, such control could be used to space the distance between tool paths and the sequence of mesh points can be used for tool moves.

Acknowledgement

This work was performed partially when the author was working at NASA Ames Research Center. This work was supported in part by DOE Contract #W-7405-ENG-48. The author would like to thank members of the PMesh Group at Lawrence Livermore Laboratories for their help and support in the course of this work.

References

[1] S E Rogers, H V Cao and T Y Su, Grid Generation for Complex High-Lift Configurations, 29th AIAA Fluid Dynamics Conference, June 15-18, 1998, Albuquerque, NM.
[2] S P Spekreuse, Elliptic Grid Generation Based on Laplace Equations and Algebraic Transformations, J of Computational Physics, 118, pp 38-61 (1995).
[3] A Khamayseh and B Hamann, Elliptic Grid Generation Using NURBS Surfaces, Computer Aided Geometric Design, 13, pp 369-386 (1996).
[4] M Mantyla, *An Introduction to Solid Modeling*, Rockville, MD (1988).
[5] L Piegl and W Tiller, Parametrization for Surface Fitting in Reverse Engineering, CAD, 33, pp 593-603 (2001).
[6] F Lin and W Hewitt, Expressing Coons-Gordon Surfaces as NURBS, CAD, 26, pp 145-155 (1994).
[7] T Y Yu and B K Soni, Application of NURBS in Numerical Grid Generation, CAD, 27, pp 147-157 (1995).
[8] J F Thompson, Z U A Warsi, and C W Martin, *Numerical Grid Generation*, North Holland, New York, NY, (1985).
[9] J F Thompson, B K Soni, N P Weatherill, *Handbook of Grid Generation*, CRC Press, New York, (1999).
[10] J C Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth and Brooks, Pacific Grove, CA, (1989).